Санкт–Петербургский государственный университет

Елизарова Ульяна Юрьевна

Выпускная квалификационная работа

Стилометрия русских текстов с помощью предобученных языковых моделей

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2017 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Математическое и программное обеспечение вычислительных машин»

Научный руководитель:
ст. преподаватель, кафедра технологий программирования
Мишенин Алексей Николаевич

Рецензент:
Юршина Анастасия Александровна

Санкт-Петербург
2021 г.

Содержание

Введение

Глобализация и повсеместная автоматизация сильно изменили мир за последние десятелетия. Каждую секунду, по данным 2020 года, среднестатистический пользователь создавал 1,7 Мб информации. Подобную ситуацию некоторые эксперты называют "информационным ожирением". В таких условиях всё большую ценность приобретают различные способы обработки информации и получения из неё максимума полезных знаний. Отдельную нишу занимает обработка естественного языка.

Обработка естественного языка (Natural Language Processing, NLP) — пересечение машинного обучения и математической лингвистики. Сегодня NLP применяется во многих сферах, в том числе в голосовых помощниках, автоматических переводах текста и фильтрации текста. Основными тремя направлениями являются: распознавание речи (Speech Recognition), понимание естественного языка (Natural Language Understanding) и генерация естественного языка (Natural Language Generation).

Частным случаем NLU является атрибуция авторства, т.е. исследование с целью установления автора текста или получения каких-либо сведений об авторе. Развитие данного направления, вкупе со стилометрическими приёмами, оказывает огромное влияние на различные области, такие как: 1. Анализ социальных сетей, политические исследования и маркетинг - для лучшего понимания аудитории, а также для изучения поведения различных политических групп довольно часто прибегают к методам атрибуции авторства [1, 2]; 2. Работа с преступлениями, связанными с кибербезопасностью, - стилометрические методы использовались в качестве доказательства в виде экспертных знаний в судах Великобритании, США и Австралии; 3. Образование и литературоведение - многие методы атрибуции авторства были созданы с целью разрешения споров о принадлежности текстов тому или иному историческому лицу[3, 4].

Данная работа представляет из себя реализацию методов атрибуции авторства на корпусе художественных текстов русских авторов.

Постановка задачи

В качестве хорошей аналогии для определения истинного автора является определение человека по отпечаткам пальцев. Последнее имеет название дактилоскопии, и начинает свою историю с конца 19го века, когда Ф. Гальтон на основе коллекции Парижской полиции (генетические тесты). Но с полной уверенностью можно утверждать, что задача атрибуции авторства сильно сложнее задачи сопоставления человека с его отпечатками.

Во-первых, стиль автора - сущность динамичная и периодически изменяющаяся, в отличие от тех же статичных отпечатков, которые являются уникальным идентификатором конкретного человека на всю жизнь (не рассматриваем случаи хируррргического вмешания или прочие крайние ситуации).

Во-вторых, задачи, связанные с работой с естественными языками, всегда сопряжены с большой трудностью - сбор и предварительная обработка данных. Количество сырых данных велико, но они редко приносят пользу в своём естественном виде. В среднем, предобработка занимает в районе 3-4 этапов (нормализация, удаление стоп-слов, токенизация и др.)

В рамках проблемы атрибуции авторство было решено использовать компьютерные методы обработки естественного языка, позволяющие по заданному художественному русскоязычному тексту определять автора из огранниченного набора альтернатив.

Сформулируем данную задачу следующим образом:

Имеется ограниченное множество авторов -

$$A = \{a_1, a_2, ..., a_k\}$$

и ограниченное множество текстов у каждого автора -

$$T_i = \left\{t_1^i, t_2^i, ..., t_l^i\right\}, i \in [1, k]$$

$$T = T_1 \cup T_2 \cup ... \cup T_i, i \in [1, k]$$

- общее множество всех доступных текстов для исследования.

Для случайно выбранного текста из множества T необходимо определить истинного автора из множества A.

В рамках работы над этим исследованием были выделены следующие этапы:

1. Изучение уже существующих статей и исследований, связанных с методами идентификации авторов русскоязычных художественных текстов;

2. Выделение наиболее перспективных приёмов и инструментов;

3. Создание для данных целей достаточно объёмного корпуса;

4. Предварительная обработка данных;

5. Программная реализация идей;

6. Проверка качества полученных результатов.

Обзор литературы

Существует довольно большое количество исследований, посвящённых анализу и идентификации авторства (включая исследования на русском языке). Среди них можно выделить три основных группы:

1. Исследования, использующие в качестве своей основы лингвистические и филологические приёмы, а также различные количественные характеристики (без использования методов машинного обучения). Чаще всего их называют "автороведческой экспертизой". В рамках подобных работ исследователи обращают внимание на особенности употребления автором знаков препинания, характерные ошибки в написании слов, особенности построения предложений и предпочтение тех или иных конструкций, словарный запас автора, а также на общую структуру написанного текста [5].

2. Исследования на основе моделей машинного обучения. Они, в свою очередь, основаны на применении методов векторизации слов и предложений, а также дальнейшем использовании моделей классификации для данных сущностей.

3. Смешанные исследования, в основе которых лежит практика по объединению методологий с применением иоделей машинного обучения и различных количественных филолого-лингвистических метрик [6].

Глава 1. Этап подготовки

Наиболее распространённым и хорошо зарекомендовавшим себя способом анализа данных является векторизация. Однако для лучшей передачи всех всей ценности изначальной информации итоговые векторы должны сохранить в себе лингвистические и стилевые особенности.

Word2Vec, Glove и другие инструменты для векторного представления отдельных слов давно и часто используются в задачах NLP. Однако, с развитием информационного общества, их возможностей начинает не хватать, поэтому появляется потребность в создании схожих моделей для более крупных структур, например, полноразмерных текстов документов или художественных произведений.

После изучения уже существующих работ и исследований по атрибуции автора для решения поставленной задачи был выбран метод Author2Vec, реализация которого не была найдена в русскоязычном интернете. Суть данного подхода заключается в векторном представлении не слов или предложений, как говорилось выше, а именно частей текстов, которые в дальнейшем используются для мультиклассовой классификации.

Рис. 1: Обзор предлагаемого решения для атрибуции авторства (источник: [11])

В качестве рабочего языка программирования был выбран Python3, как самый популярный язык для задач анализа данных и текста в частности. Из возможных сред разработок выбор пал на Google Colab — это бесплатный облачный сервис на основе Jupyter Notebook. Google Colab предоставляет всё необходимое для машинного обучения прямо в браузере, даёт бесплатный доступ к невероятно быстрым GPU и TPU.

1.1 Обзор, сбор и предобработка данных

Обзор готовых корпусов текстов на русском языке показал, что не существует достаточно крупных наборов данных с художественной литературой. Кроме того, для исследования было решено использовать произведения авторов, не носящих на себе ярлык "классиков т.к. у многих подобных личностей ярко выраженный стиль написания. Главными критериями при создании собственного корпуса были: достаточное количество различных авторов (от 100 и выше) и наличие у них нескольких (в количестве от трёх) крупных текстов (более 2000 символов).

В качестве потенциального источника данных были рассмотрены несколько вариантов:

1. Рецензии с сайта Кинопоиска. Эта возможность была отвергнута из-за отсутствия какого-либо API, а также крайне сложной структуры переходов между разделами на сервисе, не позволяющей автоматизировать сбор текстов авторов.

2. Тексты с сайта "Книга фанфиков". Здесь обнаружились проблемы, схожие с проблемами в первом пункте.

3. Тексты с сайта "Самиздат который существует с 2000 года. Именно этот вариант был выбран для работы. На момент сентября 2020 года, на данном сайте были публикации свыше 103.541 авторов. Суммарно, в разных жанрах, было опубликовано в районе 700.000 текстов, почти все на русском языке. Кроме того, структура сайта не сильно изменилась с нулевых годов, что сильно облегчило пункт с написанием парсера для выгрузки текстов.

Для автоматизации процесса выгрузки текстов был составлен список тегов, хранящих важную информацию:

Таблица 1: Таблица соответствий тегов и сущностей

Тег	Содержание
//dl/dl/dt/li/a	Текстовое содержание со страницы автора
//dd	Проза
//pre	Поэзия

В качестве формата для хранения собранных данных был выбран JSONL (JSONLINES). Каждая JSON-строка представляет из себя словарь с ключом в виде ссылки на страницу автора (все ссылки являются уникальными, в отличие от имени, поэтому они и были выбраны в качестве ключей) и значением в виде всех текстов данного автора.

JSONLINES был выбран по нескольким причинам:

1. Он компактен;

2. Его предложения легко читаются и составляются какчеловеком, так и компьютером;

3. Его легко преобразовать в структуру данных для большинства языков программирования (числа, строки, логические переменные, массивы и так далее);

4. Многие языки программирования (в частности, Python) имеют функции и библиотеки для чтения и создания структур JSON;

5. При дальнейшем анализе нет необходимости в считывании полного файла, что может оказаться проблематичным из-за большого объёма данных.

Результатом работы парсера стал файл размером 600мб.

Из-за идеи векторизации отрывков текста с сохранением специфики каждого автора было решено не удалять "стоп-слова"и знаки препинания, однако были удалены символы переноса строки и лишние пробелы. Все тексты были разделены на равные отрывки, а те, в свою очередь, на предложения с целью дальнейшего применения усреднения векторов предложений.

1.2 Описание используемой модели векторизации

В приложениях машинного перевода модели принимают на вход предложения на одном языке и выводят предложения на другом, т.е. в виде векторов. Кодирующий компонент – это стек энкодеров. Декодирующий компонент – это стек декодеров, представленных в том же количестве. Именно на них построены такие архитектуры, как траснформеры. Трансформер — это архитектура глубоких нейронных сетей, представленная в 2017 году исследователями из Google Brain.

По аналогии с рекуррентными нейронными сетями (РНС) трансформеры предназначены для обработки последовательностей, таких как текст на естественном языке, и решения таких задач как машинный перевод и автоматическое реферирование. В отличие от РНС, трансформеры не требуют обработки последовательностей по порядку. Например, если входные данные — это текст, то трансформеру не требуется обрабатывать конец текста после обработки его начала. Благодаря этому трансформеры распараллеливаются легче чем РНС и могут быть быстрее обучены.

Кодировщик трансформера получает на вход векторизованую последовательность с позиционной информацией. Декодировщик получает на вход часть этой последовательности и выход кодировщика. Кодировщик и декодировщик состоят из слоев. Слои кодировщика последовательно передают результат следующему слою в качестве его входа. Слои декодировщика последовательно передают результат следующему слою вместе с результатом кодировщика в качестве его входа.

Каждый кодировщик состоит из механизма самовнимания (вход из предыдущего слоя) и нейронной сети с прямой связью (вход из механизма самовнимания). Каждый декодировщик состоит из механизма самовнимания (вход из предыдущего слоя), механизма внимания к результатам кодирования (вход из механизма самовнимания и кодировщика) и нейронной сети с прямой связью (вход из механизма внимания).

В качестве предобученной модели для векторизации была выбрана модель на основе трансформера из-за их частой применисти к задачам NLP. LaBSE2Vec - Language-agnostic BERT Sentence Embedding. Данная

модель довольно новая, дата её реализации - 2020-07-03. Модель обучена на 17 миллиардах моноязычных предложений и 6 миллиардах биязычных пар предложений с использованием предварительного обучения MLM и TLM, в результате чего модель эффективна даже на редких языках. Именно мультиязычность стала решающим фактором, который повлиял на выбор LaBSE. Эта многоязычная модель превосходит предыдущие аналогичные, которые в основном являются двуязычными. Отличительной чертой данного инструмента являются комбинация стратегий предварительного обучения и точной настройки для повышения производительности модели с двойным кодировщиком до современной производительности в области интеллектуального анализа текстов.

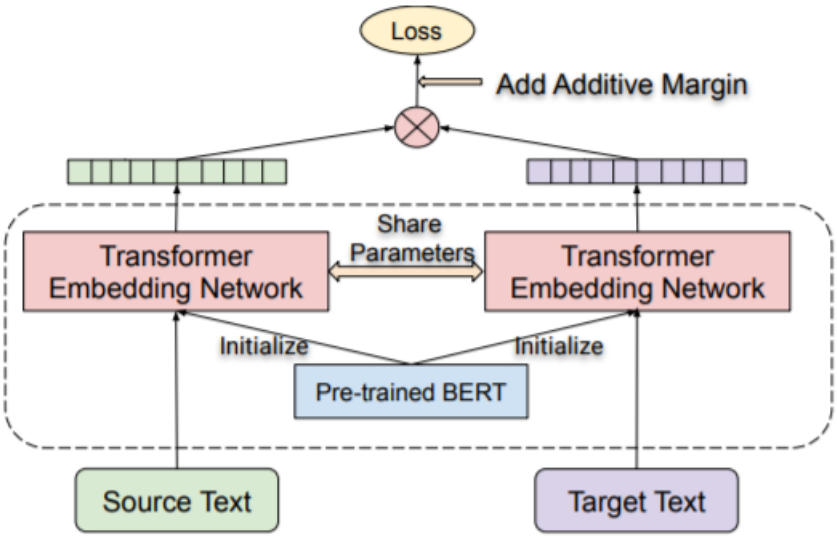

Рис. 2: Модель двойного кодировщика на основе BERT (источник: [9])

На вход данной модели подаются части текста в количестве 20, разбитые по предложениям. Полученные вектора предложений усреднялись для получения вектора одного отрывка текста.

Глава 2. Задача классификации

Задача определения по тексту автора сводится к задаче классификации с мультиклассом. Полученные векторы частей текста разбиваются в классических пропорциях на train и test выборку - 0.8 и 0.2.

2.1 Модуль CatBoost

На данном этапе была выбрана библиотека CatBoost - это библиотека градиентного бустинга, созданная Яндексом. Она использует небрежные (oblivious) деревья решений, чтобы вырастить сбалансированное дерево. Одни и те же функции используются для создания левых и правых разделений (split) на каждом уровне дерева. По сравнению с классическими деревьями, небрежные деревья более эффективны при реализации на процессоре и просты в обучении.

На сегодняшний день бустинг является одним из самых мощных алгоритмов распознавания. Это достигается благодаря адаптивной технике построения композиции. Кроме того, можно рассматривать различные функции потерь. Это позволяет решать как задачи классификации, так и задачи регрессии. К тому же, возможность выбора произвольной функции потерь позволяет акцентировать внимание на особенностях данных в задаче.

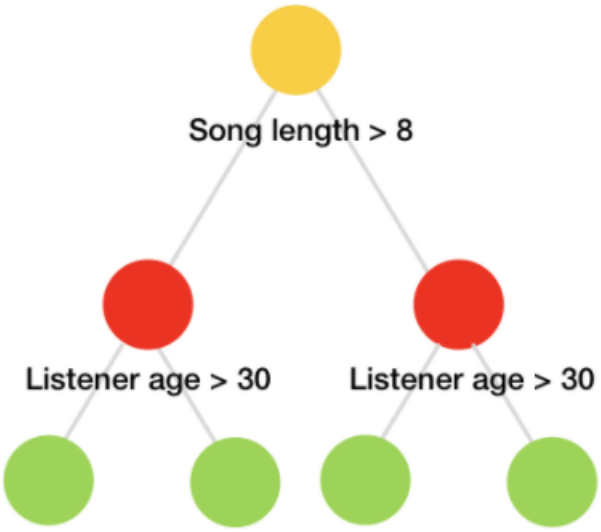

Рис. 3: Простая иллюстрация работы небрежных деревьев решений

CatBoost был выбран по нескольким причинам:

1. Библиотека позволяет получить отличные результаты с параметрами по умолчанию, что сокращает время, необходимое для настройки гиперпараметров;

2. Обеспечивает повышенную точность за счет уменьшения переобучения;

3. Возможность быстрого предсказания с применением модели CatBoost;

4. Может использоваться для регрессионных и классификационных задач.

2.2 Параметры обучения

Таблица 2: Общие параметры CatBoost

`loss_function (objective)`	Показатель, используемый для обучения. Есть регрессионные показатели, такие как среднеквадратичная ошибка для регрессии и logloss для классификации
`eval_metric`	Метрика, используемая для обнаружения переобучения
`Iterations`	Максимальное количество построенных деревьев, по умолчанию 1000. Альтернативные названия `num_boost_round`, `n_estimators` и `num_trees`
`learning_rate (eta)`	Скорость обучения, которая определяет насколько быстро или медленно модель будет учиться. Значение по умолчанию обычно равно 0.03

`random_seed (random_state)`	Случайное зерно, используемое для обучения
`l2_leaf_reg (reg_lambda)`	Коэффициент при члене регуляризации L2 функции потерь. Значение по умолчанию – 3.0
`bootstrap_type`	Определяет метод сэмплинга весов объектов, например это может быть Байес, Бернулли, многомерная случайная величина или Пуассон
`depth`	Глубина дерева

`grow_policy`	Определяет, как будет применяться жадный алгоритм поиска. Может стоять в значении SymmetricTree, Depthwise или Lossguide. По умолчанию SymmetricTree. В SymmetricTree дерево строится уровень за уровнем, пока не достигнет необходимой глубины. На каждом шаге листья с предыдущего дерева разделяются с тем же условием. При выборе параметра Depthwise дерево строится шаг за шагом, пока не достигнет необходимой глубины. Листья разделяются с использованием условия, которое приводит к лучшему уменьшению потерь. В Lossguide дерево строится по листьям до тех пор, пока не будет достигнуто заданное количество листьев. На каждом шаге разделяется нетерминальный лист с лучшим уменьшением потерь
`min_data_in_leaf`	Это минимальное количество обучающих сэмплов в листе. Этот параметр используется только с Lossguide и Depthwise.

`max_leaves (num_leaves)`	Этот параметр используется только с Lossguide и определяет количество листьев в дереве
`ignored_features`	Указывает на признаки, которые нужно игнорировать в процессе обучения
`nan_mode`	Метод работы с пропущенными значениями. Параметры Forbidden, Min и Max. При использовании Forbidden наличие пропущенных значений вызовет ошибку. При использовании параметра Min, пропущенные значения будут приняты за максимальные значения для данного признака. В Max пропущенные значения будут приняты как минимальные значения для данного признака
`leaf_estimation_backtracking`	Тип бэктрекинга, использующийся при градиентном спуске. По умолчанию используется AnyImprovement. AnyImprovement уменьшает шаг спуска до того, как значение функции потерь будет меньшим, чем оно было на последней итерации. Armijo уменьшает шаг спуска до тех пор, пока не будет выполнено условие Вольфе

boosting_type	Схема бустинга. Она может быть простой для классической схемы градиентного бустинга или упорядоченной, что обеспечит лучшее качество на небольших наборах данных
score_function	Тип оценки, используемой для выбора следующего разбиения при построении дерева. Cosine используется по умолчанию. Другие доступные варианты L2, NewtonL2 и NewtonCosine
early_stopping_rounds	Если стоит True, устанавливает тип детектора переобучения в Iter и останавливает обучение, когда достигается оптимальное значение
classes_count	Количество классов для задач мультиклассификации
task_type	Используете вы CPU или GPU. По умолчанию стоит CPU
devices	Идентификаторы устройств GPU, которые будут использоваться для обучения
cat_features	Массив с категориальными столбцами
text_features	Используется для объявления текстовых столбцов в задачах классификации

2.3 Оценка результатов модели

В качестве метрик для оценки результатов отработанных моделей была выбрана точность, т.е. средний процент правильно опознанных текстов для всех авторов в тестовой выборке.

Кроме того, в качестве очевидной гипотезы было решено проверить влияние количества авторов и длины текстовых отрывков в обучающей выборке.

Основные выводы, полученные в ходе выполнения работы:

1. Реализация архитектуры Author2Vec применима к русскому языку и показывает схожие с английским языком результаты.

2. Выдвинутые в гипотезе параметры действительно влияют на результаты: с увеличением числа авторов понижается точность модели, но с увеличением длины текстов - повышается.

В таблице нижен представлена иллюстрация данных результатов.

Таблица 3: Таблица сравнительных результатов классификации

Длина текстов	Кол.во авторов	Кол.во train-текстов	Точность
$>$ 15000	5	$>$ 4	0.88-1
>7500	5	$>$ 4	0.81 - 0.98
1000 - 5000	5	25 - 30	0.89 - 0.95
1000 - 5000	5 - 10	25 - 30	0.8 - 0.93
1000 - 5000	20 - 25	30 - 40	0.67 - 0.78

Заключение

Результатом данного исследования стало написание программы с использованием ЯП Python3 и специализированных библиотек для машинного обучения (таких как: catboost, vectorhub[encoders-text-tfhub] и другие). Идейно реализация была разбита на две части - парсер и работа с моделями.

Парсерная часть содержит в себе логику с http-запросами и реализацию класса Author, с такими атрибутами как authorName и authorTexts.

Логическая часть посвящена применению LaBSE к полученному корпусу текстов с целью их векторизации, а также применению модели классификации от catboost к train-выборке. Завершается данный этап использованием test-выборки для проверки качества работы выбранной архитектуры.

Как показали результаты, использование структуры Author2Vec, основанной на применении BERT-образных моделей, хорошо показывает себя на ранее тяжело контролируемой задаче - атрибуция авторства.

Список литературы

[1] Lau J. H., Collier N., Baldwin T. On-line trend analysis with topic models: twitter trends detection topic model online. Proceedings of COLING: Technical Papers. Mumbai, 2012, pp. 1519–1534;

[2] Е. А. Малютин, Д. Ю. Бугайченко, А. Н. Мишенин, “Выделение текстовых трендов в социальной сети ОК”, Вестн. С.-Петербург. ун-та. Сер. 10. Прикл. матем. Информ. Проц. упр., 13:3 (2017), 313–325;

[3] `https://sysblok.ru/philology/`;

[4] Дроздова, И. И. Определение авторства текста по частотным характеристикам / И. И. Дроздова, А. Д. Обухова. — Текст : непосредственный // Технические науки в России и за рубежом : материалы VII Междунар. науч. конф. (г. Москва, ноябрь 2017 г.). — Москва : Буки-Веди, 2017. — С. 18-21. — URL: https://moluch.ru/conf/tech/archive/286/13237/;

[5] GRACHEVA A.A.1. STYLOMETRY: A COMPUTER METHOD FOR ATTRIBUTION AND STYLISTIC ANALYSIS. Higher School of Printing and Media Technologies, Saint-Petersburg State University of Industrial Technologies and Design;

[6] MB Malyutov. Authorship attribution of texts: A review. General Theory of Information Transfer and Combinatorics, 362-380;

[7] `https://catboost.ai/docs/concepts/`;

[8] `https://vector-ai.github.io/vectorhub/`;

[9] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, Wei Wang. Language-agnostic BERT Sentence Embedding. Google AI;

[10] Xiaodong Wu, Weizhe Lin, Zhilin Wang, Elena Rastorgueva. Author2Vec: A Framework for Generating User Embedding. University of Cambridge, United Kingdom;

[11] STEVEN H. H., BENJAMIN C. M. FUNG, FARKHUND IQBAL, WILLIAM K. CHEUNG. Learning Stylometric Representations for Authorship Analysis;

[12] https://nplus1.ru/material/2020/10/15/pseudotragics.